Docket No. 006570.P011
Express Mail No. EV339918993US

UNITED STATES PATENT APPLICATION

FOR

SYSTEM AND METHOD FOR MANAGING COMMUNICATION BETWEEN SERVER NODES CONTAINED WITHIN A CLUSTERED ENVIRONMENT

Inventor:

Frank Kilian

Prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, LLP
12400 Wilshire Boulevard, Seventh Floor
Los Angeles, California 90025-1030
(310) 207-3800

SYSTEM AND METHOD FOR MANAGING COMMUNICATION BETWEEN SERVER NODES CONTAINED WITHIN A CLUSTERED ENVIRONMENT

BACKGROUND

Field

[0001] Embodiments of the invention relate to managing communication between multiple server nodes contained within a clustered environment.

Background

[0002] A clustered system may include a collection of servers and other components that are arranged to cooperatively perform computer-implemented tasks, such as providing client computers with access to a set of services and resources. The clustered system may be used in an enterprise software environment to handle a number of tasks in parallel. Typically, load balancing algorithm is implemented within the cluster to distribute incoming requests from the client computers evenly among multiple server nodes. In some cases, a single server node in the cluster may handle requests from a client computer. In other cases, the requests received from a client computer are handled by more than one server node cooperating with one another to perform the requested tasks.

[0003] One aspect of a clustered system is scalability. The system has the flexibility to enable adding cluster elements to the clustered system. In some cases, it may be desirable to incorporate new cluster elements that use recently developed hardware and/or software technologies. In such cases, a problem may arise if the communication protocol used by preexisting cluster elements (e.g., preexisting servers) is different from the communication protocol used by the recently added cluster elements. For example, if preexisting servers within a clustered system cannot effectively communicate with recently added servers, the preexisting and recently added servers may not be capable of cooperatively handling requests received from client computers.

SUMMARY

[0004] In accordance with one embodiment of the invention, a method and a corresponding system are disclosed for managing communication between multiple instances contained within a clustered environment. The method includes generating a packet to be transmitted from a non-Java-based server node to a Java-based server node. The non-Java-based server node specifies in a header of the packet [1] a destination server node and [2] information to indicate that the packet originated from the non-Java-based server node. The method further includes forwarding the packet from the non-Java-based server node to an intermediate server. Once the intermediate server receives the packet, the intermediate server examines the header of the packet and forwards the packet to the Java-based server node based on the destination information provided in the header of the packet.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The invention is illustrated by way of example and not by way of limitation in the figures of the accompanying drawings in which like references indicate similar elements. It should be noted that the references to "an" or "one" embodiment of this disclosure are not necessarily to the same embodiment, and such references mean at least one.

[0006] **Figure 1** shows a simplified representation of a clustered system coupled to client computers through a network according one embodiment of the invention.

[0007] **Figure 2** shows a block diagram illustrating internal components of instances contained within a clustered system according to one embodiment of the invention.

[0008] **Figure 3** shows a block diagram of a message server according to one embodiment of the invention.

[0009] **Figure 4** shows a flowchart diagram illustrating operations involved in transmitting a packet from a non-Java-based server node to a Java-based server node according to one embodiment of the invention.

[00010] **Figure 5** shows a flowchart diagram illustrating operations involved in transmitting a packet from a Java-based server node to a non-Java-based server node according to one embodiment of the invention.

DETAILED DESCRIPTION

[00011] In the following description, specific details are set forth. However, it is understood that embodiments of the invention may be practiced without these specific details. In other instances, well-known hardware and software components, structures, techniques and methods have not been shown in detail to avoid obscuring the understanding of this description.

[00012] **Figure 1** shows a simplified representation of a clustered system 100 coupled to client computers 102-1 through 102-N via a network 108 according to one embodiment of the invention. The clustered system 100 includes a set of instances 112-1 through 112-N that are coupled together through a central communication node 116. Each node, including the instances 112-1 through 112-N and the central communication node 116, contained within the clustered system 100 may include any number of server nodes. The server nodes contained within the instances 112-1 through 112-N may communicate with each other by sending packets to perform tasks requested by the client computers 102.

[00013] The clustered system 100 may be used to provide a scalable server environment that permits substantially real-time access to information for a distributed user base. In one embodiment, a dispatcher 110 is coupled between the network 108 and the instances 112-1 through 112-N to distribute requests from client computers 102 based on the load on the respective instances 112-1 through 112-N. The dispatcher may be, for example, a web dispatcher coupled to network 108 and web server nodes. The network 108 may be a local area network, a wide area network, the Internet, or any combination thereof. The network 108 may employ any type of wired or wireless communication channels capable of establishing communication between computing devices.

[00014] The instances 112-1 through 112-N contained within the clustered system may communicate with each other through the central communication node 116. The instances 112-1 and 112-N are coupled to one or more storage systems 118, 120. The storage systems 118, 120 provide storage for code and data that are used by the instances.

[00015] The clustered system 100 may include a number of Java instances running on a hardware system and a number of non-Java instances running on another hardware system. In one embodiment, the Java instances and the non-Java instances utilize separate storage systems. For example, one storage system 118 may coupled to the non-Java instances to provide storage for the non-Java instances and the other storage system 120 may be coupled to the Java instances to provide storage for the Java instances. Each instance (non-Java and Java) have one or more work processes, which are under control of a control logic executed within the corresponding instance. The control logic may be different for non-Java and Java instances.

[00016] **Figure 2** shows internal components of instances 112-1 and 112-2 according to one embodiment of the invention. Each instance includes two or more server nodes and a local dispatcher node, which dispatches requests from client computers across the multiple server nodes. In the illustrated embodiment, the first instance 112-1 includes non-Java-based server nodes 202-1 through 202-N. In one embodiment, the non-Java-based server nodes 202-1 through 202-N implement the Advanced Business Application Program (ABAP) platform. ABAP is a programming language for developing applications for the SAP R/3 system. The second instance 112-2 includes Java-based server nodes 210-1 through 210-N implementing Java 2 Platform Enterprise Edition (J2EE). J2EE is a Java platform that can be used to develop, deploy and maintain enterprise applications. Incorporating both the non-Java-based instance 112-1 and the Java-based instance 112-2 within the same clustered system enables the clustered system to handle requests intended for either Java applications or non-Java applications, such as ABAP applications. Although J2EE and ABAP programming models are described herein, it should be noted that the embodiments of the invention may be implemented with other programming models, such as, for example, WebDynpro, XI (Exchange Infrastructure) and Portals.

[00017] The clustered system 100 includes storage systems 118, 120 to provide storage for code and data that are used by the instances 112-1 and 112-2, respectively. In the illustrated embodiment, the instances 112-1 and 112-2 maintain separate storage systems. However, the clustered system 100 may be configured such that two or more instances contained therein share a single

storage system. The storage systems 118, 120 are used to store a first database 250 maintained by the first instance 112-1 and a second database 252 maintained by the second instance 112-2. In the illustrated embodiment, the ABAP applications executed in the first instance 112-1 modify data in the first database 250, and the Java applications executed in the second instance 112-2 modify data in the second database 252.

[00018] Also illustrated in **Figure 2** is a central communication node 116 coupled between the first instance 112-1 and the second instance 112-2. The central communication node 116 includes an enqueue server 226 and a message server 220. The message server 220 is responsible for managing the communication between any two instances contained within the clustered system 100. In one embodiment, the non-Java instance 112-1 and the Java instance 112-2 use the message server 220 as their runtime repository and messaging system. The enqueue server 226 is responsible for providing central locking services in order to lock accesses to resources within the clustered system 100.

[00019] The message server 220 includes a first communication interface 222 to establish communication with the non-Java-based server nodes 202-1 through 202-N of the first instance 112-1, and a second communication interface 224 to establish communication with the Java-based server nodes 210-1 through 210-N of the second instance 112-2. In one embodiment, the communication between the non-Java-based server nodes 202 and the Java-based server nodes 210 is accomplished by synchronizing the packet header between packets generated by Java applications and packets generated by ABAP applications, in order to provide compatibility. In one embodiment, a Java library contained within each Java-based server node is configured to attach a header to a body of a packet that can be decoded by the non-Java-based instances (e.g., ABAP platform) and by a protocol layer of the message server 220.

[00020] **Figure 3** shows a simplified representation of a message server 220 according to one embodiment of the invention. In one embodiment, the message server 220 provides an infrastructure for data exchange between server nodes contained within a clustered system by maintaining a list of services (e.g., processes and tasks) performed by the server nodes. The message server 220 includes a first repository 302 to maintain a list of services performed by the non-Java-based server nodes. The first repository 302 includes a number of entry rows 310-1 through 310-N, each entry row used to store information relating to a service performed by a non-Java-based server node. In the illustrated embodiment, each entry row 310 of the first repository 302 is associated with [1] a Non-Java-Based Server ID column 304 to store the server identification number associated with a respective server node performing the service, [2] a Service Mask column 306 to identify the type of service, which is represented by a bit mask in one embodiment, and [3] a Status column 308 to contain information relating to the status of the service.

[00021] Similarly, the message server 220 further includes a second repository 320 to maintain a list of services performed by the Java-based server nodes. The second repository 320 includes a number of entry rows 328-1 through 329-N, each entry row used to store information relating to a service performed by a Java-based server node. In the illustrated embodiment, each entry row 328 of the second repository 320 is associated with [1] a Java-Based Server ID column 322 to store the server identification number associated with a respective server node performing the service, [2] a Service Mask column 324 to identify the type of service, and [3] a Status column 326 to contain information relating to the status of the service. The status column may contain data relating to the status of the services, such as starting, running, terminating, etc. In one context, the term "services" is used to describe any service, process, transaction or task performed by a server node or any suitable computing device.

[00022] In operation, the message server 220 receives status information from all of the server nodes contained within the clustered system and uses the status information to update its first and second repositories 302 and 320. Each of the non-Java-based server nodes 202-1 through 202-N periodically sends a packet indicating its status (running, stopped, etc), which may include other information such as access point, port address, etc to the message server. In response, the message server 220 updates its first repository 302 based on the information contained in the status packet. The message server 220 is also configured to send notification of events that arise in the clustered system. In one embodiment, the message server sends notification of the status of all of the server nodes contained within the clustered system, for example, to indicate if a particular server node has started, shut-down or restarted.

[00023] Additionally, the message server is configured to notify the status of all of the services performed by the server nodes. Specifically, the message server 220 uses the information contained in the first repository 302 to send notification of a status of each of the listed services to the first instance containing the non-Java-based server nodes. Based on the notifications, server nodes within an instance can determine which services have failed before completion and which services are available on which server nodes. Similarly, the message server 220 uses the information contained in the second repository 320 to send notification of a status of each of the listed services to the Java-based server nodes included in another instance. In one embodiment, the information contained in the first and second repositories 302 and 320 and 320 is also supplied to a central dispatcher, which uses this information to balance the load among various instances contained within the cluster.

[00024] In one embodiment, the message server 220 includes a controller 340, which is configured to dynamically assign service identifications associated with services performed by the server nodes. By dynamically assigning service identification during runtime, the message server 220 is able to accommodate new services and increases the flexibility of handling various sets of services performed by the clustered system. The message server 220 includes a service repository 330 to maintain a list of assigned service identifications and their corresponding service names. The service repository 330 includes a number of entry rows 336-1 through 336-N, each entry row used to store a service identification (referred to in Figure 3 as an "Assigned Service Mask") associated with a particular service. In the illustrated embodiment, each entry row 336 of the service repository 330 is associated with [1] a Service Name column to store the name of a respective service, and [2] an Assigned Service Mask column to store identification of the respective service, which is represented by bit masks in one embodiment. In use, each time a service, which is not listed in the service repository 330, is performed by one of the server nodes, the controller 340 assigns a unique service mask for the corresponding service. Subsequently, whenever the same type of service is performed by one of the server nodes, it is identified by the same service mask assigned in the service repository 330.

[00025] **Figure 4** shows general operations involved in transmitting a packet from a non-Java-based server node to a Java-based server node according to one embodiment of the invention. In block 410, a non-Java-based server node generates a packet to be transmitted to one of Java-based server nodes contained within a clustered system. The packet to be transmitted may be a message packet, a request packet or an acknowledgement, the packet including a header and a body. The header is attached to the body by a communication interface coupled to the non-Java-based server node. In one embodiment, the communication interface establishes communication with other servers by using C-libraries to implement a Java-native communication protocol.

[00026] In one embodiment, the packets originating from the non-Java-based server nodes specify in the header section that the packets were transmitted from the non-Java-based server nodes. In this regard, the communication interface of the non-Java-based server node configures the header section of the packet such that it includes, among other things, a first field that specifies the address of the destination server node and a second field that specifies the packet originated from one of the non-Java-based server nodes, in block 420. In one implementation, the second field of the header indicates that the packet originated from one of the server nodes implementing ABAP applications. Then, in block 430, the non-Java-based server node forwards the packet to message server 220 in the clustered system. Once the message server has received a packet, the protocol layer included in the message server is configured to route the packet to one of the Java-based server nodes based on the destination information included in the header of the packet.

[00027] **Figure 5** shows general operations for transmitting a packet from a Java-based server node to a non-Java-based server node according to one embodiment of the invention. In block 510, a Java-based server node generates a packet to be transmitted to one of non-Java-based server nodes contained within a clustered system. The packet to be transmitted may be a message packet, a request packet or an acknowledgement packet. The packets each

includes a header and a body. The header is attached to the body by a communication interface implemented within the Java-based server node. In one embodiment, the communication interface is implemented using a Java library.

[00028] In one embodiment, the packets originating from the Java-based server nodes specify in the header section that the packets were transmitted from the Java-based server nodes. In this regard, the communication interface of a Java-based server node configures the header section of the packet such that it includes, among other things, a first field that specifies the address of the destination server node and a second field that specifies the packet originated from one of the Java-based server nodes, in block 520. Then, in block 530, the Java-based server node forward the packet to a message server 220 in the clustered system. Once the message server has received a packet, the protocol layer included in the message server is configured to route the packet to one of the non-Java-based server nodes based on the destination information included in the header of the packet.

[00029] In one embodiment, a single centralized message server is used to coordinate processes executed in non-Java-based server nodes as well as the Java-based server nodes. This may be accomplished by synchronizing the packet header utilized by all server nodes included in the clustered system. In one embodiment, the packet headers generated by ABAP applications are compatible with the header of packets generated by Java application programs.

[00030] In one embodiment, a failover mechanism is implemented by the message server. This is accomplished by returning to the message server a confirmation for each message sent by the message server, the confirmation indicating that the message has been received by a component of the clustered system and sent to the corresponding recipient. In case a component of the clustered system fails and prevents the message from being sent to the recipient, the message server will know that the message has not been properly processed by virtue of not receiving a confirmation and will send the message again when the failed component is up and running. By implementing such a failover mechanism, there is a greater likelihood that the message transmitted by the message server will be processed and will not be lost.

[00031] While the invention has been described in terms of several embodiments, those skilled in the art will recognize that the invention is not limited to the embodiments described, but can be practiced with modification and alteration within the spirit and scope of the appended claims. The description is thus to be regarded as illustrative instead of limiting.